記事
Toshihiko Minamoto · 2021年10月7日 5m read

ローカル変数スコープの概要

これは、オンラインドキュメントのさまざまな場所に表示される主題に関する概要であり、主に注釈として表示され、専用の章として表示されることはありません。

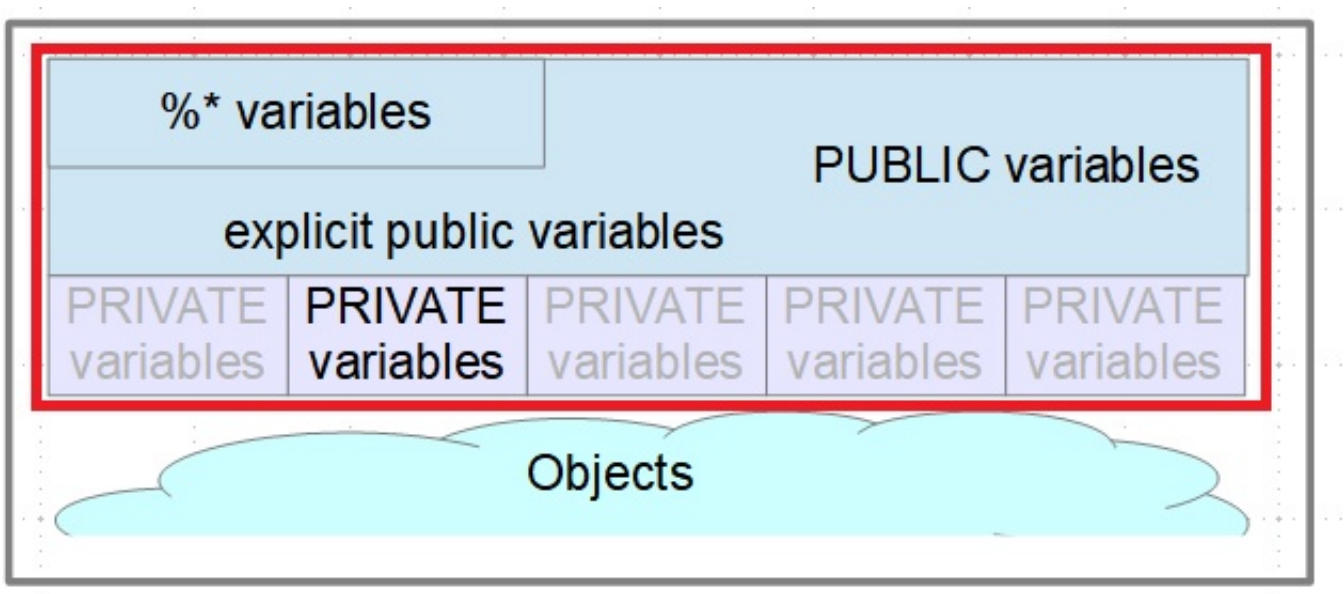
むかしむかし、ある所に... おっと、おとぎ話ではありません。
Cachéの初めの頃（それからその前にも）、自分のコードを実行するためのパーティションを用意していたことがあるでしょう。
そのパーティションの一部は、すべてのローカル変数が%、A～Z、a～zでうまくソートされていた領域でした。

また、ローカルに保存する値や情報が何であれ、すべてはそこに保存されており、パーティションでどんなコードを実行する場合でも、可視状態であり、利用することができました。
完全なドキュメントと優良な規律をもって共同作業できている開発者チームであれば、問題はありません。
［残念ながら、これをおとぎ話にしてはいけません］。

実際に、（独自または外部の）ソフトウェアパッケージを使って作業することは、当時は悪夢であり、コードそのものよりも競合しない変数の使用方法を見つけることに労力を要していました。
言うまでもなく、有意義な命名は例外になっていたのです。 その頃助けとなっていたのは、
スタックに変数をプッシュして後で復元する**NEW**コマンドです。
https://docs.intersystems.com/ens20181j/csp/docbook/DocBook.UI.Page.cls?KEY=RCOScnew

Caché 5.0（2002）での実際のソリューションは、これらのコードブロック専用の変数に、空の独立した領域を使用して、プロシージャとメソッドに別々のスコープを作成することでした。
変数の誤使用や同一名の使用から保護することが困難でなくなりました。
クラスとメソッドの定義において、パラメーターの名前は**ProcedurBlock**になったのです。
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=ROBJmethodprocedureblock
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=GORIENTclassmethodkeywords

変数の従来のスコープは**PUBLIC**でしたが、新しいスコープは**PRIVATE**になりました。
ただし、基本的なルールがあります。**%** ****で始まるものはすべてPUBLIC変数で、
そのほかには明示的な設定が必要というルールです。



キーワード ProcedureBlock（デフォルト=1）は、変数のPRIVATEスコープが利用できるかどうかを制御します。
見事な改善です！
使いやすい関数を実装することで、プロシージャはスコープの設定でより単純になったのです。
また、オブジェクトクラスでのメソッドの実装が、はるかに分かりやすくなっています。
今では、すべてのPUBLIC変数に並行して、大規模なセットのPRIVATE変数を持つことが可能になったのです。
（異様な名前を変数に付ける言い訳がなくなりました！）わかりやすく言うと、一連のPRIVATE変数に対し、映画『ハイランダー』の次の原則が適用されます。
「生き残れるのはただ一人！」 http://wiki.c2.com/?HighlanderPrinciple

ある時点で、1セットのPublic変数と1セットのPrivate変数がアクティブであるということです。

ただし、実際にはちょっとした例外があります。*ByReference*
という変数を呼び出されたメソッド/プロシージャに渡す場合、呼び出し元のスコープに戻る小さな扉ができます。

ProcedureBlock=0 またはNot
ProcedureBlock
というバリアントは主に旧式のロジックに依存するレガシーコードにターゲットされていました。
ただし、たまに、生成されたコードには、それを依然として必要または使用する影の部分があります。

一般的に、変数スコープにおける競合は稀です。
2つの**例外**と1つの回避策があります。
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=GCOSusercodeindirxecjob

Xecuteは、基本的な定義でパブリック変数に対してのみ機能します。
プロシージャの呼び出しにそれとなく似てパラメータを渡すことのできる程度です。
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=RCOScxecute
この方向での更なる拡張は、**$Xecute()** 関数によって行われます。
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=RCOSfxecute

間接参照は、パブリック変数でのみ機能します。 回避策はありません。
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=GCOSoperators#GCOSoperatorsindirection
間接参照に手を出さないようにするもう一つの理由と言えるでしょう。

変数スコープに関するその他の関連ドキュメントはこちらをご覧ください。
https://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=GCOSvariables
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=GORIENTchcos#GORIENTcosscope
http://docs.intersystems.com/latestj/csp/docbook/DocBook.UI.Page.cls?KEY=GCOSusercode#GCOSusercode_argsbyref

#初心者 #Caché #InterSystems IRIS

ソースURL:
https://jp.community.intersystems.com/post/%E3%83%AD%E3%83%BC%E3%82%AB%E3%83%AB%E5%A4%89%E6%95%B0%E3%82%B9%E3%82%B3%E3%83%BC%E3%83%97%E3%81%AE%E6%A6%82%E8%A6%81